MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
DEPARTAMENTO DE TELEMÁTICA

RELATÓRIO ATIVIDADE 1 2024.1

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA DESENHAR FORMAS GEOMÉTRICAS EM UM PLANO BIDIMENSIONAL NORMALIZADO.

AVALIAÇÃO DA EFICÁCIA DE ALGORITMOS DE RASTERIZAÇÃO USANDO VARIÁVEIS TESTE EM RESOLUÇÕES DEFINIDAS

Relatório da Atividade 1 para disciplina de Computação Gráfica parte das avaliações da N1, sob orientação do Prof. Dr. Ajalmar Rego da Rocha Neto

João Victor Ipirajá e Marcelo de Araújo
Campus Fortaleza
Agosto / 2024

RESUMO

Este relatório constituímos uma abordagem abrangente para a rasterização de elementos gráficos bidimensionais em $\mathbb{R}^2$, abordando segmentos de retas, curvas de Hermite e polígonos convexos. Os algoritmos propostos são fundamentados em sólidos princípios teóricos, combinando conceitos matemáticos com técnicas de computação gráfica. O algoritmo de rasterização de segmentos de retas, é adaptado para lidar com uma variedade de situações, destacando-se sua robustez e eficiência, pois conforme (DUDA; HART, 1972; GONZALEZ; WOODS, 2008; ROSENFELD, 1966), a representação de um segmento de reta em uma matriz é uma técnica comumente utilizada em processamento de imagem. . Em seguida, as curvas de Hermite são exploradas, demonstrando sua implementação por meio da geração de segmentos de retas entre pontos de controle. Por fim, a técnica de rasterização de polígonos convexos é desenvolvida, utilizando o algoritmo de rasterização de segmentos de retas em conjunto com o método scanline, com o objetivo de produzir e analisar polígonos convexos para diversas resoluções, avaliando a qualidade dos resultados obtidos. a representação de um segmento de reta em uma matriz binária é uma técnica comumente utilizada em processamento de imagem.

SUMÁRIO

Sumário **3**
1 Introdução **4**
2 Fundamentação **5**
2.1 Pré-Processamento 6
2.1.1 Funcionalidades Auxiliares 6
2.1.1.1 Normalização dos Pontos 7
2.1.1.2 Conversão de Coordenadas 7
2.1.2 Objetos para o Mundo 8
2.1.2.1 Geração de Matrizes 2D 8
2.1.2.2 Aplicação e o Posicionamento dos Objetos no Plano Cartesiano 9
3 Metodologia **10**
3.1 Rasterização de Segmentos de Reta 10
3.2 Interpoloção de Hermite 11
3.3 Rasterização de Poligonos 13
3.4 Técnicas de Recorte 15
3.4.1 Recorte de Retas e Curvas 15
3.4.2 Recorte de Polígonos 16
3.5 Rasterização de Objetos 16
4 Resultados e Discussões **17**
4.1 Rasterização de Segmentos de Reta 18
4.2 Rasterização de Curvas 23
4.3 Rasterização de Polígonos 26
4.4 Recorte 29
4.5 Sistema de Coordenadas do Mundo 29
5 Conclusões **30**
6 Referências **32**
. . . . . **32**

1 INTRODUÇÃO

Conforme (AZEVEDO; CONCI, 2003), o advento das Unidades de Processamento Gráfico (GPUs) revolucionou o processamento de gráficos, com a capacidade de executar operações em larga escala sobre vértices, primitivas e fragmentos. A evolução desses hardwares permitiu a programação de tarefas específicas do pipeline gráfico, como a primeira etapa, por meio de shaders. Os shaders, como os vertex shaders, possibilitam o processamento independente de cada vértice, o que é aproveitado para a execução paralela em múltiplos vértices. Assim, os desenvolvedores têm controle total sobre as transformações geométricas nessa etapa do pipeline.

Além dos vertex shaders, as GPUs modernas oferecem tessellation shaders, permitindo a personalização da subdivisão de primitivas gráficas. Essa capacidade é fundamental para lidar com polígonos não convexos e adaptar malhas de forma eficiente. Entender a distinção entre polígonos convexos e não convexos é crucial, uma vez que afeta diretamente o processo de rasterização. Um polígono é considerado convexo se qualquer segmento de reta traçado entre dois pontos de seu interior permanecer dentro dele, caso contrário, é classificado como não convexo.

Segundo (GOLUB; LOAN, 2013), a álgebra linear desempenha um papel crucial na representação e transformação de imagens em computação gráfica. Um dos conceitos fundamentais é o espaço vetorial, que consiste em uma coleção de vetores que obedecem a propriedades matemáticas específicas, como a adição e a multiplicação por escalar. Esses espaços são essenciais para representar pontos no espaço tridimensional e realizar operações de transformação. Nesse contexto, a rasterização é fundamental na síntese de imagens, pois converte representações vetoriais contínuas em representações matriciais discretas.

2 FUNDAMENTAÇÃO

Inicialmente, abordamos a rasterização de retas, onde uma reta contínua, definida por (x_1, y_1) e (x_2, y_2), é aproximada na grade de pixels do dispositivo. O algoritmo determina os pixels que devem representar a reta com base na equação $y = mx + b$. Dependendo da relação entre $|\Delta x|$ e $|\Delta y|$, a rasterização pode ser feita por coluna ou linha, ou, no caso de retas diagonais ($|\Delta x| = |\Delta y|$), por qualquer uma dessas abordagens.

Avançamos para a rasterização de curvas de Hermite, definidas por pontos extremos P_1 e P_2, e vetores de tangentes T_1 e T_2. As curvas são parametrizadas pelos polinômios de Hermite de terceira ordem:

$$H_0(t) = 2t^3 - 3t^2 + 1 \tag{1}$$

$$H_1(t) = -2t^3 + 3t^2 \tag{2}$$

$$H_2(t) = t^3 - 2t^2 \tag{3}$$

$$H_3(t) = t^3 - t^2 \tag{4}$$

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} G \tag{5}$$

Também implementamos a rasterização de polígonos utilizando o algoritmo de varredura de linha (Scanline), que calcula as interseções das arestas com cada linha de varredura, determinando os pixels internos ao polígono.

Além disso, abordamos algoritmos de recorte como Cohen-Sutherland e Sutherland-Hodgman, essenciais para delimitar e extrair regiões específicas em cenas tridimensionais ou imagens bidimensionais, otimizando a renderização e manipulação de objetos gráficos.

Nosso objetivo é desenvolver algoritmos eficientes para a rasterização de segmentos de retas, curvas e polígonos no plano bidimensional $\mathbb{R}^2$. Conforme descrito por (FOLEY *et al.*, 1990), a rasterização converte uma descrição geométrica em uma imagem de pixels. Implementamos o algoritmo do ponto médio para rasterização de semirretas, curvas de Hermite para geração de curvas suaves, e o algoritmo de varredura de linha para polígonos.

Esses algoritmos fornecem uma base robusta para a visualização e análise de gráficos vetoriais, convertendo gráficos vetoriais em representações matriciais, essenciais para exibição em dispositivos de saída. A rasterização de segmentos de reta foi implementada de forma recursiva para tratar diferentes inclinações, incluindo casos especiais de retas verticais e horizontais, avaliando o desempenho em termos de robustez e eficiência, conforme os requisitos do projeto e as resoluções de imagem. A equação da reta e o critério de decisão do algoritmo do ponto médio são expressos como:

$$y = mx + b$$

$$d = x_{\text{start}} - x_{\text{end}} + 2(y_{\text{start}} - y_{\text{end}})$$

Os pixels são selecionados de acordo com a atualização do critério de decisão:

$$d = d + 2(y_{\text{start}} - y_{\text{end}})$$

ou

$$d = d + 2(y_{\text{start}} - y_{\text{end}}) - 2(x_{\text{end}} - x_{\text{start}})$$

O algoritmo de blending de Hermite rasteriza curvas usando os pontos de controle e vetores tangentes, enquanto o algoritmo scanline varre os eixos, interpolando linearmente as interseções para determinar os pixels internos ao polígono.

Por fim, os algoritmos de recorte e rasterização implementados são fundamentais para a manipulação e visualização eficiente de elementos geométricos em diversos contextos de computação gráfica, permitindo recortar objetos fora da janela de visualização e rasterizar apenas as porções visíveis.

2.1 Pré-Processamento

Neste trabalho, propomos uma abordagem estruturada para a manipulação e transformação de objetos geométricos no plano cartesiano, utilizando operações matriciais e técnicas de normalização. Cada uma dessas etapas foi desenvolvida com o objetivo de fornecer uma metodologia robusta e flexível para a manipulação de objetos geométricos no plano cartesiano, contribuindo para uma representação gráfica precisa e consistente no contexto de aplicações computacionais.

2.1.1 Funcionalidades Auxiliares

A manipulação e transformação de coordenadas são aspectos fundamentais em diversas áreas da computação gráfica e da modelagem geométrica, permitindo a representação, normalização e rasterização de objetos em espaços bidimensionais e tridimensionais. No presente estudo, abordamos a normalização dos pontos e a conversão de coordenadas como passos críticos para garantir a consistência e a precisão das operações subsequentes, como transformações geométricas e rasterização. Essas técnicas, quando combinadas, formam a base para a manipulação precisa e eficiente de objetos geométricos em sistemas de coordenadas, permitindo a aplicação de transformações e rasterização com alta precisão.

2.1.1.1 Normalização dos Pontos

No estudo, implementamos uma abordagem de normalização dos pontos, que segundo (RUDIN, 1987), em um espaço normalizado, também conhecido como espaço com norma, as três propriedades fundamentais são satisfeitas: positividade, escalabilidade e a desigualdade triangular. Utilizamos uma classe específica para tal, sendo chamada pelo método estático na classe de rasterizar o mundo, ajustando os valores dos pontos para um intervalo padrão, entre 0 e 1. Isso garante que as coordenadas dos pontos fiquem dentro de uma escala uniforme. Dado um ponto ou uma lista de pontos p, o processo de normalização segue os seguintes passos:

Calculamos os valores mínimo e máximo para o ponto p em cada dimensão; min_val = $\min(p)$ e max_val = $\max(p)$. Logo após, cada coordenada do ponto p é normalizada usando a seguinte fórmula:

$$p_{\text{normalizado}} = \frac{p - \text{min\_val}}{\text{max\_val} - \text{min\_val}}$$

Isso garante que os valores normalizados estarão dentro do intervalo $[0, 1]$, onde o valor 0 corresponde ao valor mínimo $min(p)$ e o valor 1 corresponde ao valor máximo $max(p)$. Esse processo é essencial para garantir que as operações subsequentes, como transformações geométricas e/ou rasterização, sejam realizadas de forma consistente.

2.1.1.2 Conversão de Coordenadas

A estrutura é projetada para manipular coordenadas e realizar transformações entre representações homogêneas e cartesianas. Nela, temos dois métodos: transformar coordenadas cartesianas em homogêneas e vice-versa. Iniciamos esses processos com a definição de um conjunto de coordenadas. No primeiro método, convertemos de coordenadas cartesianas para coordenadas homogêneas, pois, de acordo com (GIATTI, 2020), a desvantagem das coordenadas cartesianas é que elas não suportam o conceito de pontos no infinito. Esta limitação obriga os algoritmos geométricos a tratar separadamente muitos casos particulares. Em termos matemáticos, as coordenadas homogêneas adicionam uma dimensão extra ao espaço, geralmente chamada de w, o que facilita a aplicação de transformações lineares.

$$\mathbf{p} = (x, y, z) \xrightarrow{\text{Transformação}} \mathbf{p_h} = (x, y, z, 1) \xrightarrow{\text{Transposta}} \mathbf{P}^T = \begin{bmatrix} \mathbf{P_h} \\ 1 \end{bmatrix}$$

No segundo método, realizamos a conversão inversa, transformando coordenadas homogêneas de volta para coordenadas cartesianas. Isso é feito removendo a última linha das coordenadas homogêneas e transpondo a matriz resultante. Se $\mathbf{p_h} = (\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$ representa um ponto em coordenadas homogêneas, sua correspondente em coordenadas cartesianas é dada por:

$$\mathbf{p} = \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$$

A remoção da última linha e a transposição são realizadas para converter as coordenadas de volta ao formato cartesiano.

$$\mathbf{p} = (\mathbf{P_h}\,[:-1,:])^T$$

A implementação fornece uma forma eficiente de alternar entre representações homogêneas e cartesianas de coordenadas, permitindo a aplicação de transformações geométricas de forma mais simplificada e flexível.

2.1.2 Objetos para o Mundo

Implementamos classes e métodos responsáveis pela manipulação e rasterização de uma lista de pontos, realizando transformações e aplicando comandos específicos sobre esses pontos. Essa classe opera em várias etapas, que incluem a aplicação de transformações geométricas e o posicionamento de objetos no plano cartesiano, incluindo operações adicionais. Integramos funcionalidades de normalização e transformação para preparar os pontos de forma que possam ser convertidos em uma representação rasterizada. Para facilitar a compreensão do leitor, estruturamos a metodologia dessa parte da implementação em tópicos distintos.

2.1.2.1 Geração de Matrizes 2D

Projetamos uma classe para gerar matrizes de transformação 2D, baseadas em deslocamentos e índices fornecidos. Utilizamos essas matrizes para aplicar transformações lineares e de translação a pontos em um espaço bidimensional. Inicialmente, criamos uma matriz de transformação 3×3 que inclui escalonamento e translação, a qual empregamos para transformar coordenadas no plano cartesiano, gerando a seguinte matriz:

$$T = \begin{bmatrix} dx & 0 & dx \cdot j \\ 0 & dy & dy \cdot i \\ 0 & 0 & 1 \end{bmatrix}$$

Onde:

- dx e dy representam os fatores de escalonamento ao longo dos eixos x e y, respectivamente.
- i e j representam as translações ao longo dos eixos y e x, respectivamente.

2.1.2.2 Aplicação e o Posicionamento dos Objetos no Plano Cartesiano

O objetivo dessa etapa é transformar uma lista de objetos, aplicando matrizes de transformação a objetos geométricos representados em coordenadas homogêneas para manipulá-los de forma a serem posicionados ou distribuídos em diferentes regiões do plano cartesiano. Segundo (VINCE, 2010), o plano cartesiano fornece um mecanismo para traduzir pares de variáveis relacionadas em um formato gráfico. No contexto desta etapa, isso envolve aplicar uma série de transformações que movem e redimensionam os objetos, transformando esses objetos em novas posições e orientações no espaço bidimensional por meio de operações matriciais, posicionando-os em regiões específicas do plano, denominadas quadrantes. Cada objeto nessa lista representa um conjunto de pontos que será transformado de acordo com essas operações, utilizando a conversão de coordenadas para facilitar as operações entre as representações cartesianas e homogêneas. A principal funcionalidade desta etapa está implementada no método **apply**, que recebe dois parâmetros:

- objects: uma lista de objetos geométricos representados como listas de coordenadas.
- matrices: uma lista de matrizes de transformação que serão aplicadas a cada objeto.

Inicialmente, em uma primeira etapa, determinamos o número total de objetos a serem transformados. Em seguida, definimos a granularidade da divisão dos quadrantes, estabelecendo como os objetos serão distribuídos e transformados no plano. Os quadrantes são representados em um array, correspondendo às quatro direções principais: superior direito, superior esquerdo, inferior esquerdo e inferior direito. Para cada quadrante, a classe gera uma matriz de transformação 2D, armazenando-as em uma lista. No fim desta etapa, aplicamos as matrizes de transformação a cada objeto da lista, reposicionando-os de acordo com as transformações realizadas, e retornamos a lista de objetos transformados.

Na segunda etapa, percorremos cada objeto e sua respectiva matriz de transformação, realizando as seguintes operações de verificação: inicialmente, verificamos se o objeto é uma lista de sub-objetos (ou pontos) e se contém pelo menos dois elementos. Caso positivo, aplicamos a transformação a cada sub-objeto individualmente. Caso contrário, a transformação é aplicada diretamente ao objeto. Em seguida, o objeto ou sub-objeto é convertido para coordenadas homogêneas, permitindo a aplicação de transformações lineares. Logo após, aplicamos a matriz de transformação por meio da multiplicação matricial, conforme ilustrado abaixo.

$$T' = M \times T$$

Onde:

- T representa a matriz de coordenadas homogêneas do objeto.

- M é a matriz de transformação.
- T' é a nova matriz de coordenadas homogêneas após a transformação.

Após aplicarmos a transformação, o objeto resultante é convertido de volta para coordenadas cartesianas, removendo a dimensão extra adicionada anteriormente. Em seguida, o objeto transformado é armazenado em uma lista. Esse processo nos permite aplicar transformações complexas, como rotações, escalas e translações, de maneira eficiente e estruturada.

3 METODOLOGIA

A metodologia que adotamos fundamenta-se na utilização e aplicação de algoritmos de rasterização de retas, curvas e polígonos; juntamente com tecnicas de recorte seguindo uma abordagem sistemática para investigar a aplicação de tais algoritmos. A construção do estudo foi organizada em múltiplas fases metodológicas, uma prática consolidada e bem fundamentada na pesquisa científica, especialmente nas áreas de computação gráfica e análise de dados. Esta abordagem sistemática assegura uma organização clara e sequencial do processo de pesquisa, garantindo que cada fase seja meticulosamente planejada e executada. Conforme discutido por (CRESWELL, 2014), uma estrutura metodológica bem definida permite maior rigor e reprodutibilidade na pesquisa. Adicionalmente, (YIN, 2018) enfatiza que a segmentação do estudo em etapas distintas facilita a análise e a interpretação dos resultados. Além disso, tal estrutura facilita a replicação do estudo e a verificação dos resultados por parte de outros pesquisadores, contribuindo para a robustez e a transparência científica.

3.1 Rasterização de Segmentos de Reta

A rasterização de linhas é uma técnica fundamental em gráficos computacionais, onde se convertem representações matemáticas de linhas em pixels ou fragmentos para serem exibidos em dispositivos de visualização. Conforme (TANG; WU; ZHANG, 1992), uma linha reta é renderizada como uma sequência de pixels em displays raster. Devido à resolução finita dos dispositivos raster, erros de quantização são inevitáveis na renderização, muitas vezes causando linhas digitais irregulares visualmente questionáveis. Em nosso estudo visamos discretizar uma linha representada por um conjunto de pontos no espaço bidimensional, gerando um número específico de fragmentos que representam essa linha de forma aproximada. Tal tecnica é particularmente útil em simulações gráficas e renderização de linhas.

O script desenvolvido recebe como entrada um array de pontos, onde cada ponto é definido por suas coordenadas no plano 2D. utilizamos na construção da função um parâmetro que determina o número de fragmentos a serem gerados entre dois pontos consecutivos. A função opera de forma recursiva quando o número de pontos é maior que dois, dividindo o problema em

subproblemas menores até que reste apenas uma linha a ser rasterizada entre dois pontos. Essa abordagem divide a linha em segmentos menores, que são então rasterizados individualmente.

Dado um conjunto de pontos $\mathbf{P} = \{P_0, P_1, \ldots, P_{m-1}\}$, onde cada ponto $P_i = (x_i, y_i)$ é definido por suas coordenadas no plano 2D, a função de rasterização de uma linha entre dois pontos P_0 e P_1 pode ser expressa da seguinte maneira:

$$\Delta x = x_1 - x_0$$

$$\Delta y = y_1 - y_0$$

O incremento para cada fragmento ao longo do eixo x e y é dado por:

$$\text{incrementos} = \left(\frac{\Delta x}{n}, \frac{\Delta y}{n}\right)$$

onde n é o número de fragmentos.

As coordenadas do i-ésimo fragmento ao longo da linha são dadas por:

$$P_i = P_0 + i \times \text{incrementos}$$

Explicitamente:

$$P_i = \left(x_0 + i \times \frac{\Delta x}{n}, y_0 + i \times \frac{\Delta y}{n}\right)$$

onde i varia de 0 a n.

Se houver mais de dois pontos, a função é aplicada recursivamente para cada par de pontos consecutivos P_i e P_{i+1} no conjunto $\mathbf{P}$. O resultado de cada aplicação é armazenado em uma lista

$$\text{resultados} = \{\text{rasterizar\_linha}(P_i \cup P_{i+1}, n) \mid i = 0, 1, \ldots, m-2\}$$

onde $\cup$ representa a concatenação dos pontos P_i e P_{i+1} em um único vetor, e n é o número de fragmentos.

3.2 Interpoloção de Hermite

A interpolação de Hermite é uma técnica utilizada em gráficos computacionais para gerar curvas suaves que passam por um conjunto de pontos de controle, levando em conta tanto a posição quanto a derivada (ou tangente) nos pontos de controle. Segundo (JÚNIOR, 1996) curvas cúbicas paramétricas são aquelas cujas componentes $x(t)$ e $y(t)$ são equações

polinomiais cúbicas. Em nosso estudo combinamos a interpolação de Hermite para gerar pontos ao longo de uma curva, seguida da rasterização desses pontos para criar um conjunto de pixels que representa a curva discretizada.

Primeiramente, calculamos os pontos de uma curva de Hermite com base nos pontos de controle P e vetores tangentes T. Para um intervalo t variando de 0 a 1, a curva é avaliada utilizando a matriz de Hermite. Dados os pontos de controle P e os vetores tangentes T, a curva de Hermite é calculada para um conjunto de n pontos ao longo do parâmetro t.

$$t_i = \frac{i}{n-1}, \quad \text{onde } i = 0, 1, 2, \ldots, n-1$$

A matriz de potência T para a curva de Hermite é:

$$T = \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

A matriz de Hermite H é definida como:

$$H = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

A matriz geométrica G que combina os pontos de controle e os vetores tangentes é dada por:

$$G = \begin{pmatrix} P_0 & P_1 \\ T_0 & T_1 \end{pmatrix}$$

Finalmente, os pontos ao longo da curva são calculados por:

$$\text{pontos} = T^T \cdot H \cdot G$$

Logo após, construímos um script que gera segmentos de curva de Hermite entre pares consecutivos de pontos de controle, acumulando esses segmentos para formar a curva completa. A função itera sobre pares consecutivos de pontos de controle para calcular os segmentos da curva. Cada segmento é calculado usando a função anteriormente descrita.

$$\text{segmentos} = C(t; P_{\text{start}}, P_{\text{end}}, n)$$

Onde:

C(t; P_{start}, P_{end}, n) representa os pontos ao longo da curva de Hermite entre P_{start} e P_{end} para n pontos;

T(t) é a matriz de potência que depende de t;

H é a matriz de Hermite;

e G(P_{start}, P_{end}) é a matriz geométrica construída a partir dos pontos de controle P_{start} e P_{end}

Finalmente, os pontos calculados são rasterizados utilizando a funcão de rasterizar segmentos de retas para criar uma representação discreta retornando um array contendo os pontos rasterizados, que são a representação da curva no espaço bidimensional. Uma peculiaridade importante desta função em relação ao algoritmo clássico é a utilização da rasterização de segmentos de retas para calcular as representações discretas entre os pontos da interpolação.

3.3 Rasterização de Poligonos

Em nosso estudo implementamos uma abordagem para rasterizar polígonos e preencher suas áreas internas utilizando um algoritmo que encontra as interseções de um segmento de linha com um eixo x ou y , dado um valor constante no outro eixo, algoritmo de scanline que percorre linha a linha para preencher o polígono e por fim o algoritmo de rasterização de segmentos de ratas chamado para os pares de interseções, onde ela gera todos os pontos (ou pixels) entre os dois pontos de interseção. Particulamente nessa etapa do estudo enfrentamos grandes dificuldades pois segundo (JUNFU KONG WEIHUA, 2015) as eficiências computacionais dos algoritmos tradicionais de polígonos baseados em computação vetorial diminuirão rapidamente ao lidar com polígonos que contêm grande quantidade de vértices. Os fluxos de computação dos algoritmos tradicionais de polígonos são fortemente acoplados com estruturas de dados especiais, que são difíceis de serem otimizadas no subjacente a eles.

Inicialmente, projetamos uma função para encontrar as interseções de um segmento de linha com um eixo (x ou y), localizando as interseções de uma linha paralela a um eixo coordenado com as arestas do polígono. A função recebe os seguintes parâmetros:

- `x_or_y`: o valor constante no eixo em que as interseções estão sendo procuradas.
- `vertices`: uma lista de coordenadas de vértices do polígono.
- `axis`: o eixo (x ou y) ao longo do qual as interseções estão sendo procuradas.

Para cada par de vértices consecutivos p_1 e p_2, verificamos se a scanline cruza a aresta formada por esses vértices, caso ocorra uma interseção, calculamos a coordenada da interseção no eixo oposto usando a fórmula da interpolação linear abaixo:

$$\text{value} = p1_{\text{other\_axis}} + \frac{(x\_or\_y - p1_{\text{axis}}) \times (p2_{\text{other\_axis}} - p1_{\text{other\_axis}})}{p2_{\text{axis}} - p1_{\text{axis}}}$$

Onde:

- `p1_axis`: Coordenada do ponto p1 ao longo do eixo relevante
- `p1_other_axis`: Coordenada do ponto p1 ao longo do outro eixo
- `p2_axis`: Coordenada do ponto p2 ao longo do eixo relevante
- `p2_other_axis`: Coordenada do ponto p2 ao longo do outro eixo
- `x_or_y`: Valor constante ao longo do eixo relevante

Finalizando, a função retorna as interseções são ordenadas em ordem crescente.

Na varredura de linha e no preenchimento de polígonos implementamos o algoritmo scanline, técnica da varredura de linha que é comumente utilizado em gráficos computacionais para preenchimento de áreas delimitadas por polígonos. A função recebe os seguintes parâmetros:

- `polygons`: Matriz com as coordenadas dos vértices do polígono.
- `step`: Passo da varredura; define a resolução da rasterização.

Inicialmente os limites mínimo e máximo de x e y do polígono são calculados através da equação abaixo:

$$\min_x = \min(\text{polygon}[:, 0]), \quad \max_x = \max(\text{polygon}[:, 0])$$
$$\min_y = \min(\text{polygon}[:, 1]), \quad \max_y = \max(\text{polygon}[:, 1])$$

Depois iniciamos a varredura no eixo x e y onde para cada valor de x e y dentro do intervalo $[\min_x, \max_x]$ e $[\min_y, \max_y]$ com incremento do step. As interseções com as arestas do polígono são encontradas utilizando a função de encontrar as interseções de um segmento de linha com um eixo (x ou y), depois processadas em pares para preencher as linhas entre elas. Os pontos preenchidos são gerados pela rasterização de segmentos de retas, que discretiza a linha em pixels.

Assim obtemos duas matrizes de pontos, a primeira representando o contorno do polígono e a segunda o preenchimento interno do mesmo. Nossa tecnica apresenta uma peculiaridade em relação ao algoritmo clássico de scanline ao integrar uma função de rasterização

de segmentos de retas para gerar pontos de preenchimento ao longo dos segmentos horizontais entre interseções. Em vez de um passo fixo, utilizamos uma resolução elevada e adaptável (step), proporcionando uma precisão aprimorada na representação dos polígonos. O algoritmo processa interseções em pares, assegurando que as linhas entre essas interseções sejam corretamente preenchidas. Além disso, o uso de um conjunto para armazenar e remover duplicatas de pontos de contorno e a ordenação das interseções garantem a integridade e precisão dos dados. Essas modificações resultam em uma abordagem detalhada e precisa para a rasterização e preenchimento de polígonos, diferenciando-se do algoritmo clássico que pode não incluir tais funcionalidades ou ajustes de resolução.

3.4 Técnicas de Recorte

Neste tópico descrevemos a nossa implementação de algoritmos de recorte de retas, polígonos e curvas que segundo (SHIRLEY; ASHIKHMIN; MARSCHNER, 2009), o recorte (ou clipping) é uma operação fundamental em computação gráfica que envolve a seleção de partes visíveis de objetos gráficos dentro de uma região de interesse, geralmente definida por uma janela de visualização, tais algoritmos implementados proporcionam uma solução eficiente para o recorte de retas, polígonos e curvas no espaço bidimensional.

3.4.1 Recorte de Retas e Curvas

Implementamos o algoritmo Cohen-Sutherland, método clássico utilizado para o recorte de retas e curvas, ou seja, para determinar a porção de uma reta ou curva que se encontra dentro de uma janela de recorte definida por um retângulo, o mesmo funciona atribuindo códigos de posição a cada ponto da reta e da curva e utilizando operações lógicas para determinar se a reta e/ou curva é completamente visível, completamente invisível ou se precisa ser cortada seguindo a lógica abaixo:

$$\text{Code} = \begin{cases} \text{LEFT} & \text{SE } x < x_{\min} \\ \text{RIGHT} & \text{SE } x > x_{\max} \\ \text{BOTTOM} & \text{SE } y < y_{\min} \\ \text{TOP} & \text{SE } y > y_{\max} \\ \text{INSIDE} & \text{Caso Contrário} \end{cases}$$

A nossa implementação do algoritmo de recorte funciona iterativamente, verificando as condições de visibilidade para cada segmento de reta:

- Se ambos os pontos têm um código de saída **INSIDE**, o segmento inteiro está dentro da janela de recorte.

- Se a combinação dos códigos de saída dos dois pontos resulta em um código diferente de zero, a reta e/ou é completamente invisível.
- Caso contrário, a reta e/ou curva é parcialmente visível, e um ponto de interseção é calculado usando as regras abaixo:

$$x = x_0 + \frac{(x_1 - x_0) \cdot (y_{\text{edge}} - y_0)}{y_1 - y_0}$$

ou

$$y = y_0 + \frac{(y_1 - y_0) \cdot (x_{\text{edge}} - x_0)}{x_1 - x_0}$$

Onde (x_0, y_0) são os pontos extremos da reta, e $(x_{\text{edge}}, y_{\text{edge}})$ representam os limites da janela de recorte.

3.4.2 Recorte de Polígonos

Implementamos o algoritmo Sutherland-Hodgman, método clássico utilizado para o recorte de polígonos contra uma janela de recorte, o mesmo funciona iterando sobre os lados do polígono e aplicando o recorte contra cada borda da janela de recorte sequencialmente. Para cada aresta do polígono, o algoritmo verifica se os vértices estão dentro ou fora da janela de recorte seguindo a lógica abaixo:

$$\text{inside}(p, \text{edge}) = \begin{cases} p_x \geq x_{\min}, & \text{SE a borda é } left \\ p_x \leq x_{\max}, & \text{SE a borda é } right \\ p_y \geq y_{\min}, & \text{SE a borda é } bottom \\ p_y \leq y_{\max}, & \text{SE a borda é } top \end{cases}$$

Se um vértice está dentro da janela de recorte e o próximo está fora (ou vice-versa), o algoritmo calcula a interseção entre a aresta do polígono e a borda da janela usando as equações abaixo:

$$y = y_1 + \frac{(y_2 - y_1) \cdot (x_{\text{edge}} - x_1)}{x_1 - x_1}$$

ou

$$x = x_1 + \frac{(x_2 - x_1) \cdot (y_{\text{edge}} - y_1)}{y_1 - y_1}$$

Onde (x_1, y_1) e (x_2, y_2) são os vértices da aresta do polígono.

3.5 Rasterização de Objetos

Desenvolvemos esta classe visando transformar e rasterizar pontos ou objetos bidimensionais com base em comandos específicos, como transformações geométricas e recortes. Ao

inicializar um objeto dessa classe, fornecemos pontos que representam objetos ou formas no plano bidimensional. Primeiramente, normalizamos esses pontos, garantindo que todos estejam em uma faixa de valores comum, o que facilita os cálculos subsequentes e assegura consistência, independentemente da escala original. Em seguida, aplicamos as transformações aos pontos normalizados por meio de uma instância da classe de posicionamento dos objetos no plano cartesiano. Nesta etapa, os pontos normalizados são utilizados para gerar objetos transformados nos quadrantes do plano, sendo que cada objeto é manipulado conforme especificado.

Os comandos internos da classe desempenham um papel fundamental na manipulação e processamento desses pontos transformados. Desenvolvemos o método **_apply_commands**, responsável pela execução dos comandos internos, que itera sobre os comandos fornecidos e os aplica aos pontos transformados. Caso o parâmetro booleano seja definido como verdadeiro, o método verifica e ignora pontos vazios, garantindo que apenas pontos válidos sejam processados. De maneira similar, o método **_apply_clip_commands** aplica comandos de recorte aos pontos transformados, ajustando-os conforme a janela de recorte antes de retorná-los para processamento posterior.

A verificação de pontos vazios é realizada pelo método **_is_empty**, essencial para evitar a inclusão de pontos inválidos no conjunto final de dados. Por fim, o método **_stack_points** empilha os pontos processados, assegurando que o formato final dos dados seja consistente e adequado para uso posterior, seja para exibição ou operações adicionais.

Implementamos dois métodos principais para a aplicação de comandos aos objetos previamente transformados. O método **rasterize** permite que o usuário forneça comandos por meio de um dicionário, onde cada chave representa uma função a ser executada sobre os pontos, e cada valor é uma tupla que contém um intervalo de índices e uma posição específica para extrair os resultados gerados pela função. O método **rasterize_clip** é semelhante ao **rasterize**, mas inclui a funcionalidade adicional de recorte (clipping). Antes de aplicar os comandos externos, os pontos transformados são submetidos a comandos de recorte que ajustam os pontos de acordo com uma janela de recorte. Após o recorte, os comandos externos são aplicados aos pontos resultantes, finalizando o processo.

4 RESULTADOS E DISCUSSÕES

A rasterização é um processo essencial na computação gráfica, responsável por converter representações geométricas de imagens em uma grade de pixels que pode ser exibida em dispositivos de saída. Nesse tópico, abordamos a rasterização de elementos gráficos bidimensionais, com foco particular na rasterização de reta, curvas de Hermite e polígonos convexos. Também levamos em consideração o comportamento dos objetos nas diferentes resoluções proposta pelos requisitos do trabalho. As discussões subsequentes exploram não apenas os resultados práticos da implementação do algoritmo, mas também os desafios enfrentados e as soluções propostas para mitigar esses problemas. Além disso, abordamos as aplicações práticas

da rasterização em diversos contextos, visando fornecer uma visão abrangente sobre a eficácia dos algoritmos desenvolvidos, suas implicações práticas e as oportunidades para aprimoramento e inovação na área.

4.1 Rasterização de Segmentos de Reta

A implementação do nosso algoritmo de rasterização de segmentos de reta demonstrou sua capacidade de lidar com diversas inclinações de linhas, assegurando que sejam desenhadas de maneira contínua e precisa. Observamos que o algoritmo é eficiente na determinação dos pixels que devem ser ativados para representar, de forma exata, uma linha entre dois pontos específicos.

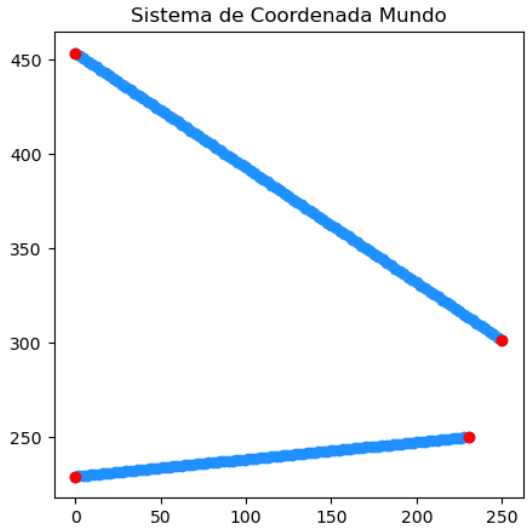

Figura 1 – Exemplos de retas

Para avaliar a eficiência e precisão do algoritmo, rasterizamos cinco segmentos de retas diferentes em quatro resoluções distintas. As resoluções foram definidas no requisito do trabalho, permitindo que o algoritmo fosse escalável para diferentes tamanhos de tela e densidades de pixels. Os cinco segmentos de reta foram escolhidos para abranger uma variedade de inclinações e posições dentro do plano cartesiano, incluindo retas com inclinações positivas e negativas, bem como segmentos verticais e horizontais.

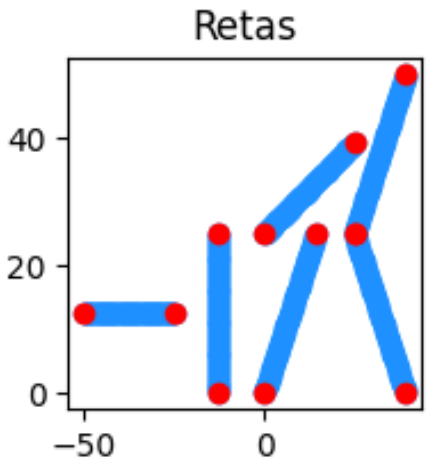

Figura 2 – Retas na resolução 100x100

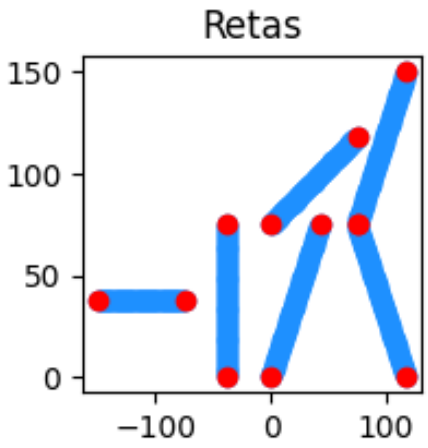

Figura 3 – Retas na resolução 300x300

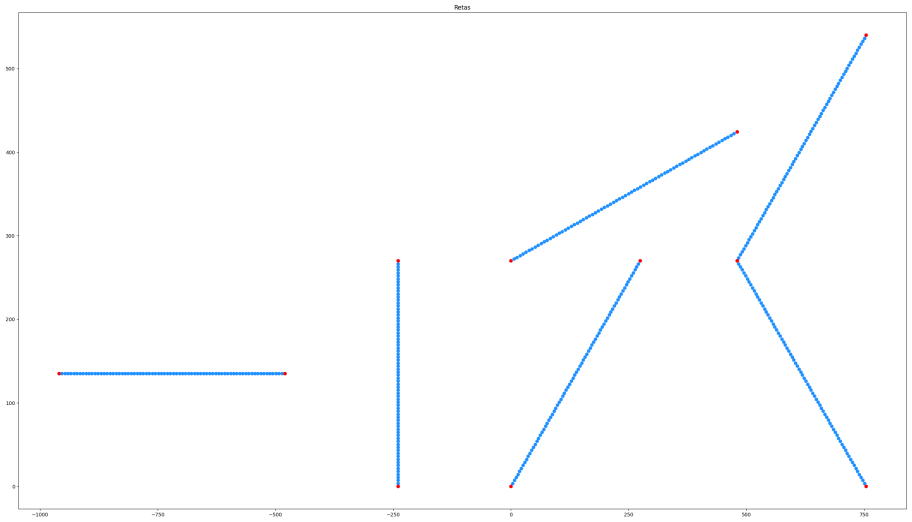

Figura 4 – Retas na resolução 1920x1080

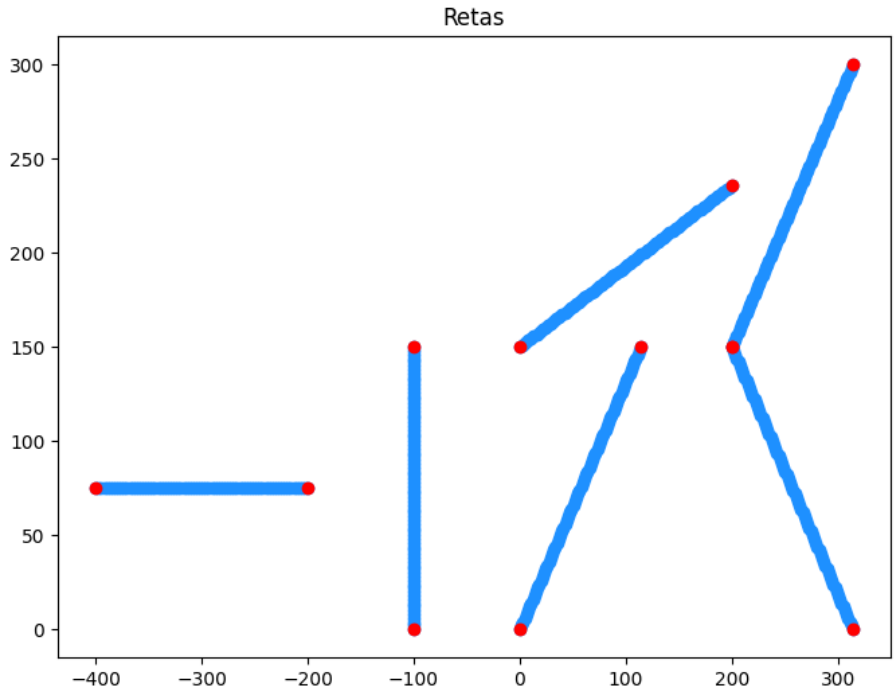

Figura 5 – Retas na resolução 800x600

Além das inclinações gerais, consideramos também situações específicas de retas verticais, horizontais, retas crescentes $m > 0$ e decrescentes $m < 0$. Para retas horizontais $m = 0$, o algoritmo foi otimizado para traçar uma linha constante no eixo y, enquanto que para retas verticais ($m \to \infty$), o traçado ocorre em uma linha constante no eixo x e para $m > 0$, o incremento ocorre na direção positiva dos eixos, enquanto para $m < 0$, o incremento no eixo y é negativo.

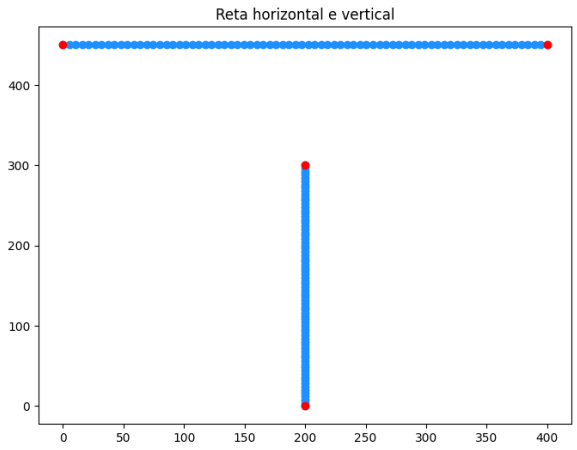

Figura 6 – Retas Horizontais e Verticais

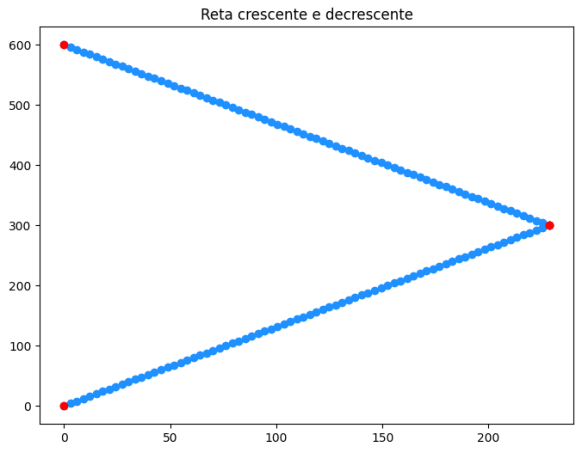

Figura 7 – Retas Crescentes e Decrescentes

Figura 8 – Retas Especificicas

O número de fragmentos é crítico para o comportamento da rasterização, pois controla a quantidade de divisões aplicadas a cada segmento da linha. Testamos a técnica com diferentes valores de fragmentos para avaliar como isso afeta a precisão e a suavidade da linha resultante. Quando utilizamos um baixo número de fragmentos, observamos que a linha rasterizada mantém uma aparência "segmentada", com poucos pontos representando a linha. Por outro lado, ao aumentar o número de fragmentos, verificamos uma melhoria significativa na suavidade da linha.

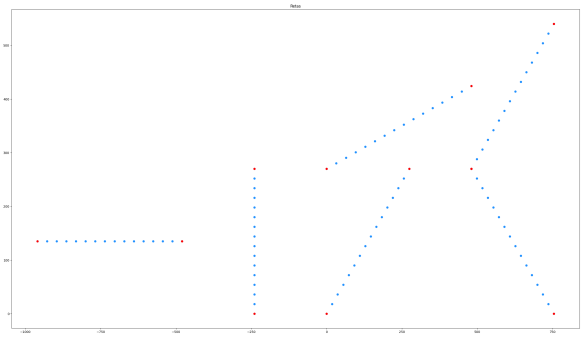

Figura 9 – Retas com 15 Fragmentos

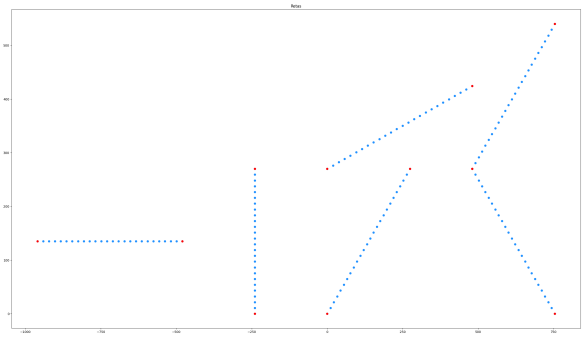

Figura 10 – Retas com 25 Fragmentos

Figura 11 – Retas com Poucos Frgmentos

Com um número maior de fragmentos, a linha é mais bem representada, pois os pontos rasterizados estão mais próximos entre si, resultando em uma visualização mais suave e contínua. Os experimentos realizados indicaram que, à medida que o número de fragmentos aumentava, a linha se aproximava cada vez mais da forma original, reduzindo a distância entre os pontos gerados e garantindo maior precisão geométrica.

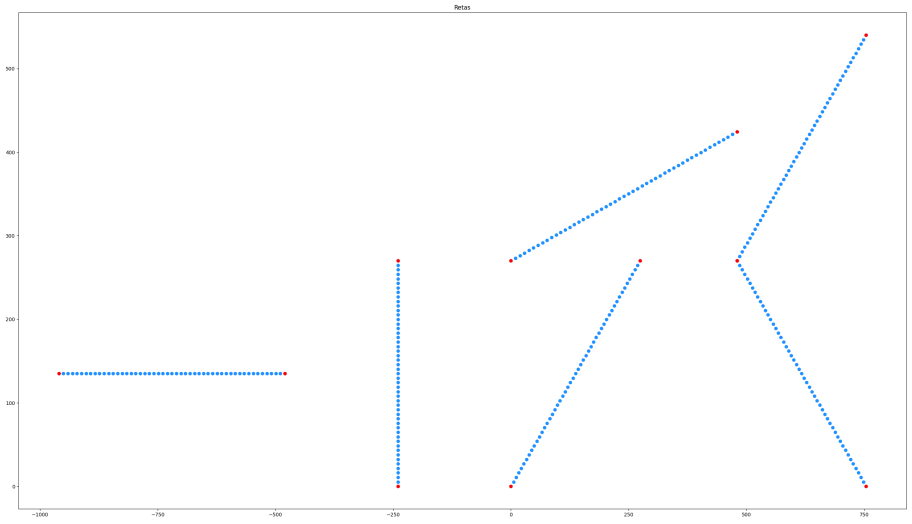

Figura 12 – Retas com 50 Fragmentos

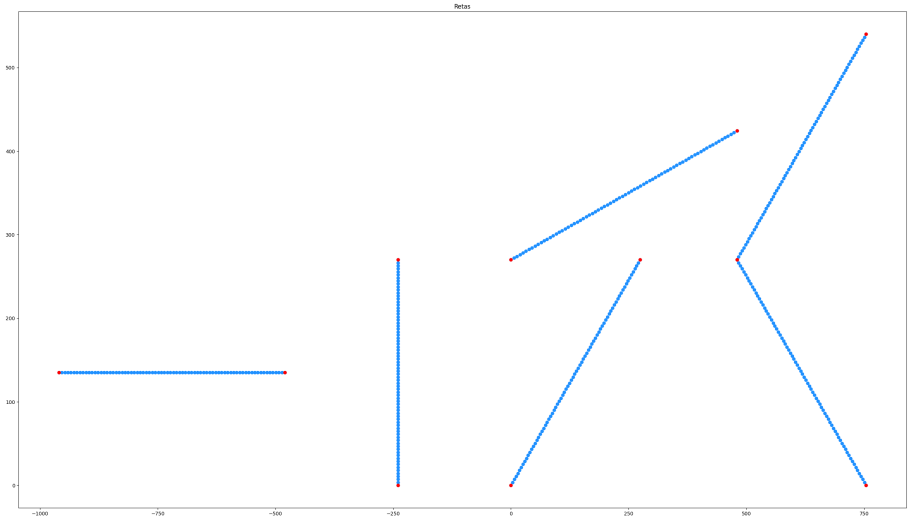

Figura 13 – Retas com 75 Fragmentos

Os resultados indicam que a versão implementada do algoritmo de rasterização de segmentos de reta é capaz de rasterizar com precisão os cinco segmentos de reta nas quatro resoluções testadas, preservando a fidelidade geométrica das linhas rasterizadas. As imagens geradas demonstram que o algoritmo mantém a integridade das retas, independentemente da resolução ou da inclinação. As situações de retas verticais e horizontais foram tratadas com precisão, assegurando que as linhas fossem traçadas de forma consistente, independentemente de suas direções. Esses resultados evidenciam a robustez, a eficiência e a eficácia da implementação.

4.2 Rasterização de Curvas

A implementação da nossa versão do algoritmo para a rasterização de curvas de Hermite utiliza um método que se baseia na definição de pontos de controle e vetores tangentes, permitindo a manipulação precisa da forma da curva. A partir dos pontos extremos P_1 e P_2, juntamente com os vetores tangentes T_1 e T_2, desenvolvemos um polinômio de terceira ordem que descreve a curva. Visualizamos que o algoritmo é eficiente na determinação dos pontos e tangentes, na geração de pontos para a curva e na aplicação da rasterização entre os mesmos pontos.

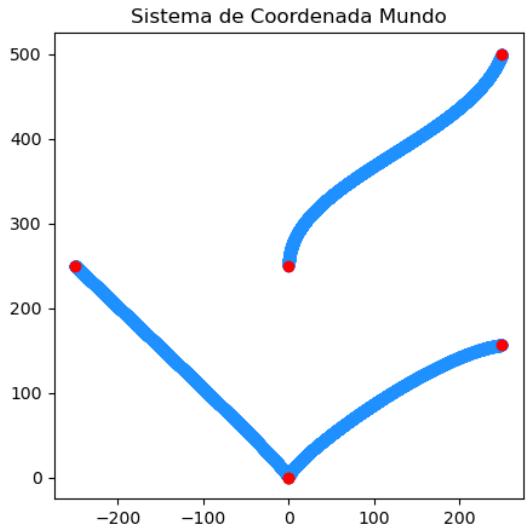

Figura 14 – Exemplos de Curvas

Para avaliar a eficiência e precisão do algoritmo implementado, rasterizamos curvas com diferentes pontos e tangentes nas resoluções definidas no requisito do trabalho, permitindo a escalabilidade para diferentes resoluções e densidades de píxeis. A escolha foi feita para cobrir uma variedade de curvas e posições dentro do espaço bidimensional, incluindo curvas com diferentes números de pontos

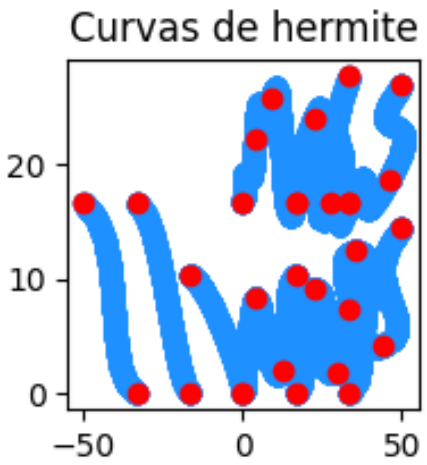

Figura 15 – Curvas em 100x100

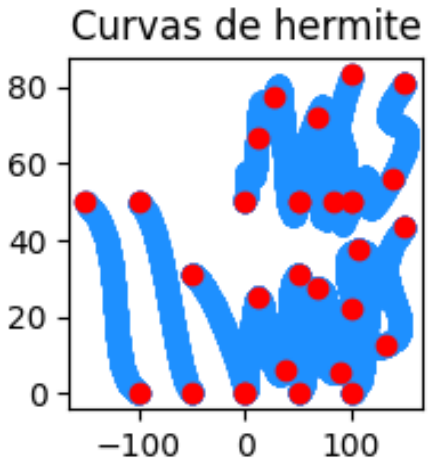

Figura 16 – Curvas em =300x300

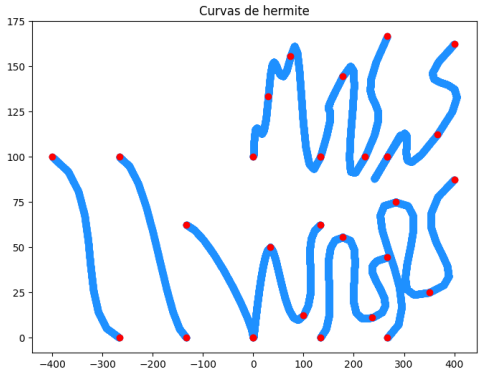

Figura 17 – Curvas em 800x600

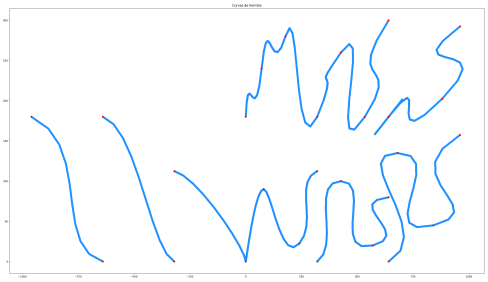

Figura 18 – Curvas em 1920x1080

Figura 19 – Curvas requiridas no trabalho

Testamos a rasterização das curvas com 3, 5 e 10 pontos e a curva tornou-se altamente flexível, permitindo a criação de formas complexas e detalhadas. A interação entre os pontos proporciona um controle refinado sobre a forma da curva, permitindo a modelagem de transições suaves e abruptas conforme necessário. A análise dos resultados indica que, à medida que o número de pontos aumentava, a capacidade de modelar formas complexas e dinâmicas também aumenta, demonstrando a versatilidade da nossa implementação do algoritmo de Hermite.

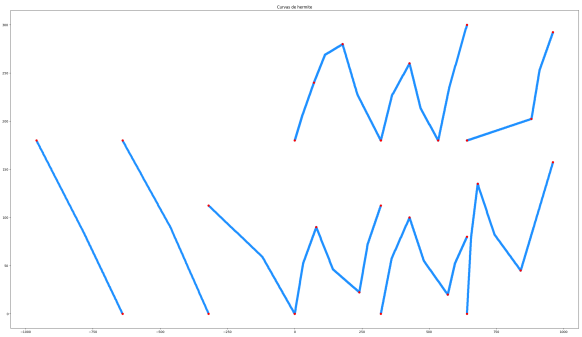

Figura 20 – Curvas de 3 pontos

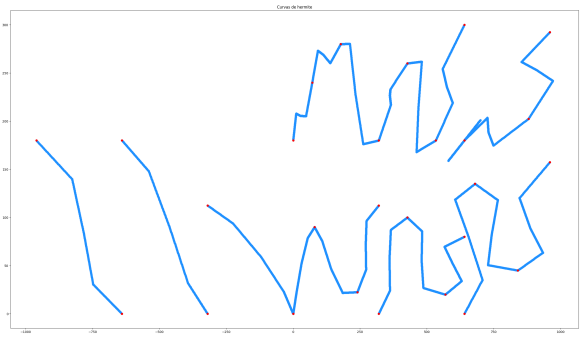

Figura 21 – Curvas de 5 pontos

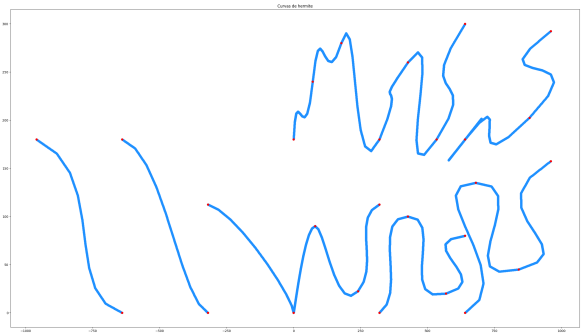

Figura 22 – Curvas de 10 pontos

Figura 23 – Curvas com diferentes números de pontos

Finalmente analisamos um experimento onde $P_1 = P_2$, nessa configuração resulta em uma curva que, em essência, se comportou como um ponto fixo, sem variação visível mostrando que a curva não tem uma transição suave entre dois pontos distintos.

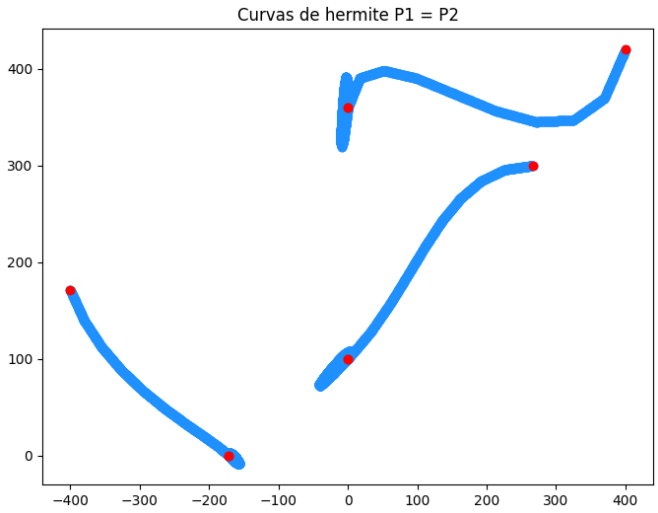

Figura 24 – Curvas com $P_1 = P_2$

Demonstramos nos resultados obtidos que a abordagem adotada para a rasterização de curvas de Hermite é eficaz na geração de representações visuais de alta qualidade. Ao implementarmos o algoritmo de Hermite, conseguimos interpolar os pontos ao longo da curva, resultando em uma transição suave entre os pontos de controle. A visualização das curvas foi realizada em diferentes resoluções, permitindo uma análise detalhada da precisão e da qualidade da rasterização.

4.3 Rasterização de Polígonos

A nossa implementação da rasterização de polígonos utilizando o algoritmo de scanline com o algoritmo de rasterizar retas foi realizada com sucesso. Testando a técnica em polígonos definidos por vértices no espaço bidimensional, o conjunto de pontos de contorno é coletado durante o processo de varredura tanto no eixo x quanto no eixo y. Esses pontos definem os limites externos do polígono, garantindo que o contorno seja preciso e delimitado corretamente no espaço 2D. Já no conjunto de pontos internos utiliza-se a rasterização linha, esse preenchimento é feito ao conectar pares de interseções detectadas durante o processo de varredura no eixo x. O algoritmo segue a lógica de preencher entre pares de interseções consecutivas, resultando em uma cobertura completa do polígono, mostrando os resultados nas 4 resoluções requisitadas no estudo.

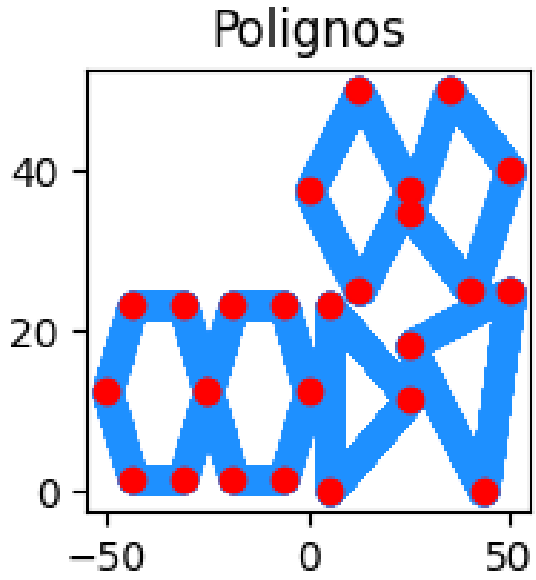

Figura 25 – Polígono Contorno 100x100

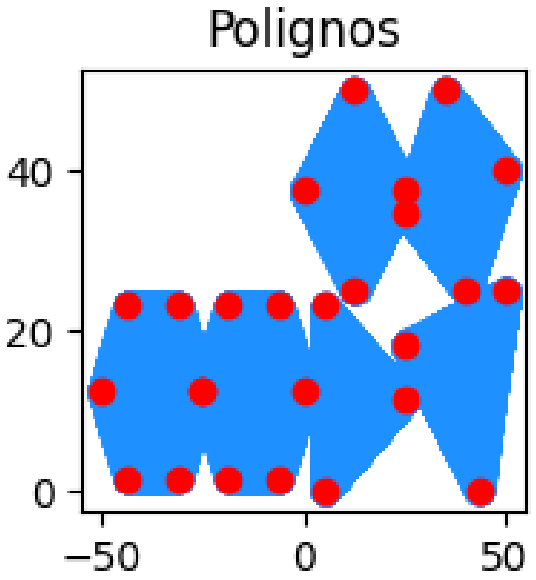

Figura 26 – Polígono Preenchido 100x100

Figura 27 – Polígonos 100x100

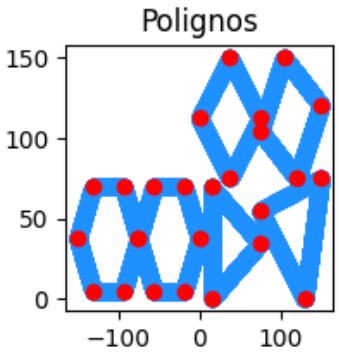

Figura 28 – Polígono Contorno 300x300

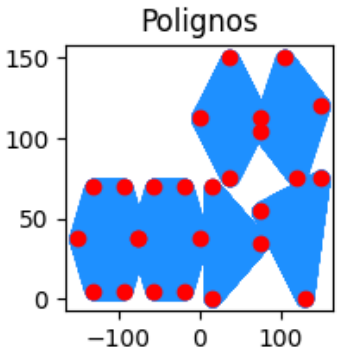

Figura 29 – Polígono Preenchido 300x300

Figura 30 – Polígonos 300x300

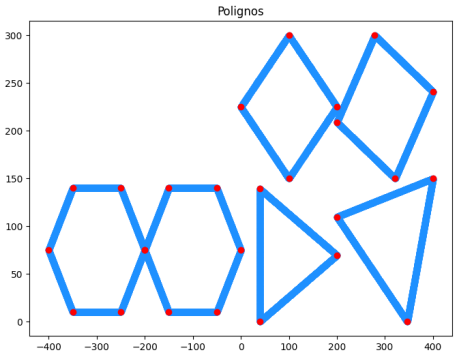

Figura 31 – Polígono Contorno 800x600

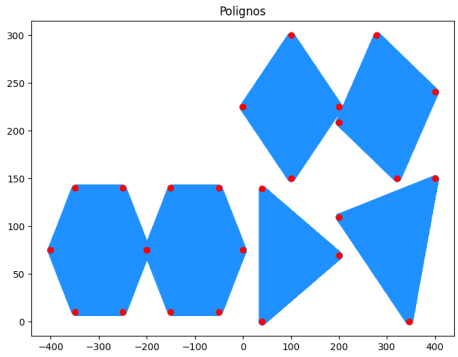

Figura 32 – Polígono Preenchido 800x600

Figura 33 – Polígonos 800x600

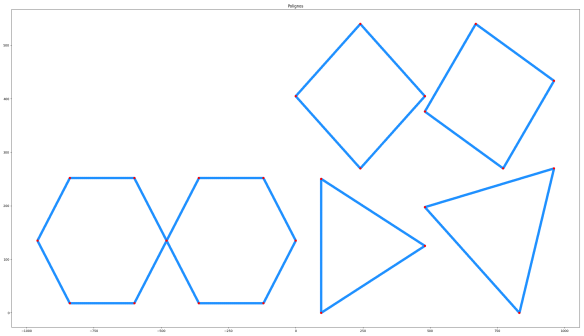

Figura 34 – Polígono Contorno 1920x1080

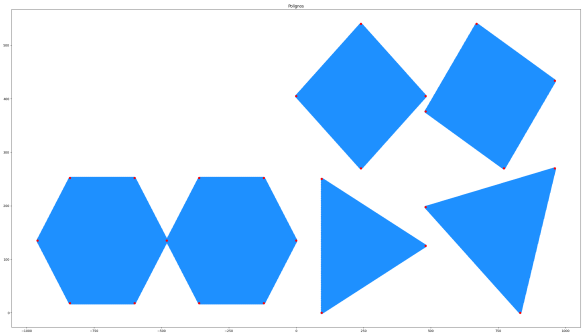

Figura 35 – Polígono Preenchido 1920x1080

Figura 36 – Polígonos 1920x1080

O parâmetro step na nossa implementação do algoritmo de scanline controla a resolução da varredura, definindo a granularidade com que a linha de varredura avança ao longo de um eixo x ou y. Quanto menor o valor do step, mais densa e precisa será a rasterização, mas isso também aumenta o custo computacional. Por outro lado, um step maior reduz o número de cálculos, porém perdem-se detalhes na forma do polígono, especialmente em bordas inclinadas ou curvas.

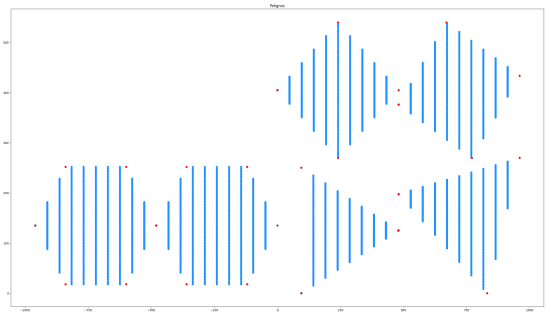

Figura 37 – Step no valor de $5e-3$

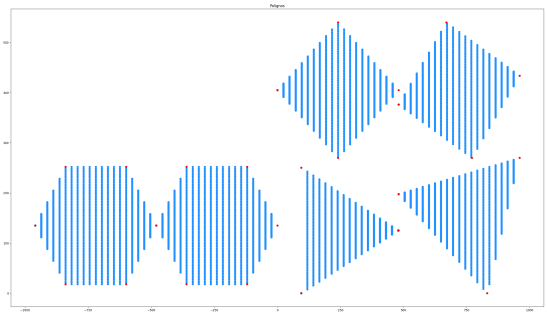

Figura 38 – Step no valor de $2.5e-3$

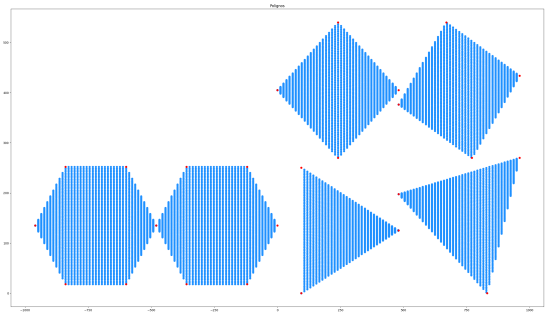

Figura 39 – Step no valor de $1.25e-3$

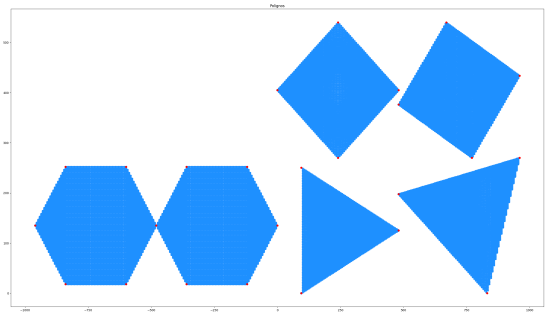

Figura 40 – Step no valor de $6.25e-4$

Figura 41 – Polígonos com diferente números de passos

Uma característica distintiva da nossa abordagem foi a integração de uma função de rasterização de segmentos de reta para gerar pontos de preenchimento ao longo dos segmentos horizontais entre as interseções. Em vez de um passo fixo, utilizamos uma resolução elevada e adaptável, o que resultou em uma precisão aprimorada na representação do espaço interno do polígono. Nossa técnica permite que o preenchimento fosse realizado de forma mais suave e contínua, evitando artefatos visuais que ocorrem em métodos de preenchimento mais simples. Em suma, a rasterização de polígonos, tanto no que diz respeito ao contorno quanto ao espaço interno, demonstra ser um processo eficaz e flexível.

4.4 Recorte

Neste estudo, implementamos algoritmos de recorte para retas, polígonos e curvas, com foco na eficiência. Para o recorte de retas e curvas implementamos o algoritmo Cohen-Sutherland, que funciona atribuindo códigos de posição a cada ponto da reta, que indicam a localização relativa do ponto em relação à janela de recorte e no recorte de curvas adicionamos de um tratamento especial para a natureza contínua das curvas, adaptando-o para lidar com a interpolação de pontos ao longo da curva.

Para o recorte de polígonos, implementamos o algoritmo de Sutherland-Hodgman processando iterativamente os vértices dos polígono e aplicando o recorte em relação a cada lado da janela de recorte, recalculando os vértices conforme necessário, mantendo a forma do polígono dentro dos limites $[-1 * aspectRatio, -1, 1 * aspectRatio, 1]$ e truncando as partes externas.

Figura 42 – Antes do Recorte

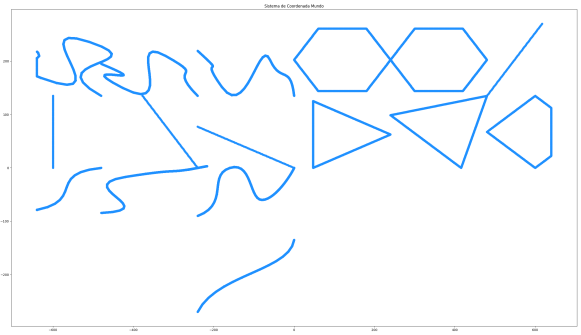

Figura 43 – Depois do Recorte

Figura 44 – Técnica de Recorte

Em conjunto, essas técnicas permitiram-nos recortar com precisão e eficiência segmentos de reta, curvas e polígonos dentro de janelas delimitadas, mantendo a integridade das formas geométricas e respeitando os limites estabelecidos pelo recorte. A nossa técnica prova ser flexível para diferentes aplicações, desde o processamento de segmentos de reta gerados por curvas interpoladas até o manejo de polígonos complexos.

4.5 Sistema de Coordenadas do Mundo

Desenvolvemos um sistema de coordenadas do mundo para rasterização de retas, polígonos e curvas, utilizando uma combinação de normalização, transformação de objetos e aplicação de comandos de rasterização e recorte. Este sistema é projetado para lidar com a representação e processamento de dados geométricos em um espaço coordenado, empregando técnicas robustas para garantir precisão e eficiência. Em termos de estabilidade e robustez, nosso sistema de coordenadas demonstra um desempenho sólido, pois a combinação de normaliza-

ção, transformações, conversão de coordenadas e rasterizações e recorte permite que o sistema gerencie eficientemente dados geométricos complexos. A capacidade de lidar com diferentes formatos de entrada e a robustez na aplicação de comandos garantem que o sistema seja aplicável a uma ampla gama de cenários e tipos de dados.

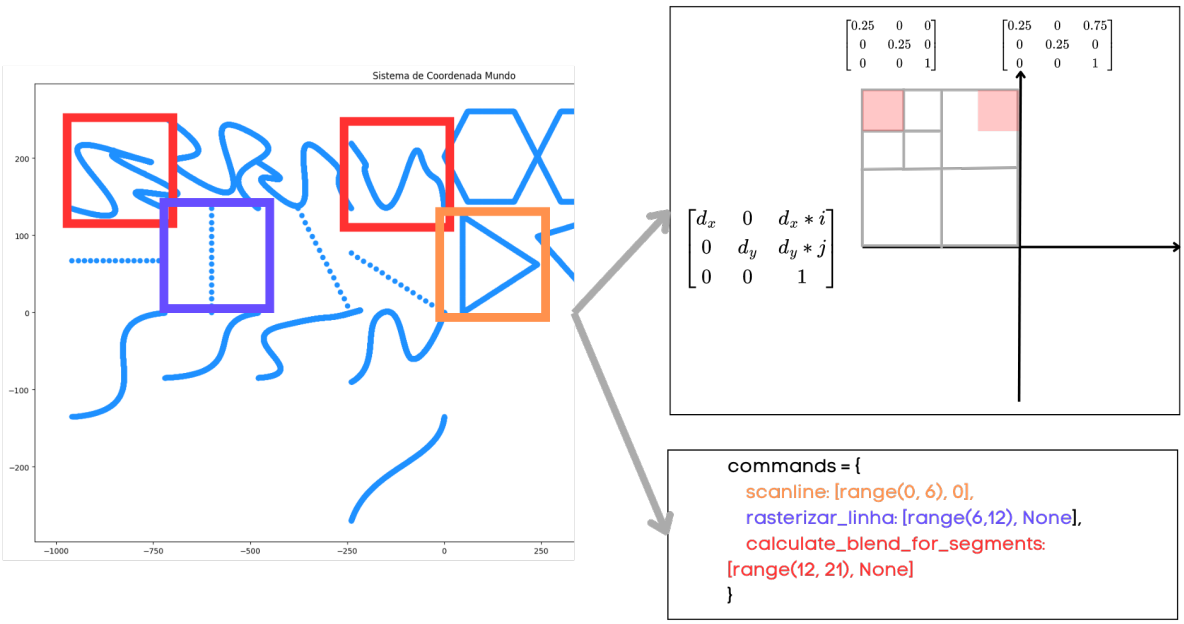

Figura 45 – Sistema de Coordenadas do Mundo

Em conclusão, o sistema de coordenadas do mundo desenvolvido mostrou-se altamente eficaz para a normalização, transformação, rasterização e recorte de pontos em um espaço coordenado. Os resultados indicam que a técnica é robusta e adequada para aplicações que exigem processamento preciso de dados geométricos em sistemas de coordenadas complexos.

5 CONCLUSÕES

As conclusões deste relatório refletem a profundidade da pesquisa e a eficácia dos métodos desenvolvidos e implementados na rasterização de elementos gráficos bidimensionais. Ao longo do desenvolvimento do nosso estudo, conseguimos integrar uma série de algoritmos que não apenas atendem às necessidades básicas de representação gráfica, mas também proporcionam uma melhoria significativa na qualidade visual e na precisão geométrica dos elementos rasterizados.

Primeiramente, ao analisarmos a rasterização de segmentos de reta, observamos que a variação no número de fragmentos utilizados tem um impacto direto na suavidade e na continuidade das linhas geradas. Mediante experimentos controlados, foi possível verificar que um aumento no número de fragmentos resulta em uma representação mais fiel da forma

original, minimizando a aparência segmentada que se observa com um número reduzido de fragmentos. Essa descoberta é crucial, pois enfatiza a importância de uma abordagem adaptativa na rasterização, que se alinha com as melhores práticas em computação gráfica.

Na implementação do algoritmo de rasterização de curvas de Hermite demonstra ser uma solução eficiente para a representação de formas curvas em um espaço bidimensional. A utilização de pontos de controle, vetores tangentes e a aplicação da rasterização de segmentos de retas entre os pontos gerados, permite uma manipulação precisa da forma da curva, resultando em uma representação visual que preserva a suavidade e a continuidade esperadas.

Além disso, a rasterização de polígonos, realizada por meio da combinação do algoritmo de rasterização de segmentos de reta e do método de varredura de linha (scanline), revelou-se uma estratégia eficaz para a representação de formas poligonais. A implementação do algoritmo de scanline permitiu a identificação precisa dos píxeis que compõem o interior dos polígonos, garantindo um preenchimento uniforme e consistente. Além disso, a implementação de uma função de rasterização adaptativa para segmentos de reta, conforme discutido, permite uma representação mais precisa do espaço interno dos polígonos. Essa técnica não apenas melhorou a suavidade do preenchimento, mas também evitou artefatos visuais indesejados que frequentemente ocorrem em métodos de preenchimento mais simples.

Sobre os algoritmos de recorte para retas, curvas e polígonos, demonstra ser eficaz na otimização da renderização. A capacidade de recortar objetos fora da janela de visualização e rasterizar apenas as porções visíveis não só melhoraram a eficiência computacional, mas também garantiu uma experiência de usuário mais fluida e responsiva.

Na construção do sistema de coordenadas do mundo foi um componente essencial para a integração e a manipulação eficaz dos elementos geométricos em nossa aplicação. Este sistema, que combina normalização, transformação de objetos e aplicação de comandos de rasterização e recorte, demonstrou uma capacidade notável de gerenciar dados geométricos complexos. A implementação de um sistema de coordenadas robusto não apenas facilitou a visualização e a interação com os elementos gráficos, mas também garantiu a precisão necessária para operações de transformação e rasterização.

Por fim, a estrutura metodológica que seguimos, fundamentada em uma abordagem sistemática, permitiu uma análise rigorosa e a replicação dos resultados. A segmentação do estudo em etapas distintas facilitou a interpretação dos dados e a verificação dos resultados, contribuindo para a robustez e a transparência científica do nosso trabalho. Em suma, as conclusões que tiramos deste relatório validam a eficácia dos algoritmos de rasterização desenvolvidos.

6 REFERÊNCIAS

AZEVEDO, E.; CONCI, A. **Computacao grafica: Teoria E pratica**. [S.l.]: Elsevier, 2003.

CRESWELL, J. W. **Research design: Qualitative, quantitative, and mixed methods approaches**. 4th. ed. [S.l.]: Sage Publications, 2014.

DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. **Communications of the ACM**, v. 15, n. 1, p. 11–15, 1972. Disponível em: <https://dl.acm.org/doi/10.1145/361237.361242>.

FOLEY, J. *et al.* **Fundamentals of Interactive Computer Graphics**. 2a edição. ed. [S.l.]: Addison-Wesley, 1990.

GIATTI, C. A. **Isometrias no R^3: modelos euclidiano e homogêneo**. Tese (Doutorado) — [sn], 2020.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. [S.l.]: Johns Hopkins University Press, 2013.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3rd. ed. Prentice Hall, 2008. Disponível em: <https://www.pearson.com/us/higher-education/program/Gonzalez-Digital-Image-Processing-3rd-Edition/PGM253813.html>.

JUNFU KONG WEIHUA, M. T. Z. C. J. M. Z. Y. F. Rapc:a rasterization-based polygon clipping algorithm and its error analysis. **Acta Geodaetica et Cartographica Sinica**, Acta Geodaetica et Cartographica Sinica, v. 44, n. 3, p. 338, 2015. Disponível em: <http://xb.chinasmp.com/EN/abstract/article_6508.shtml>.

JÚNIOR, A. d. M. B. **Aplicação dos elementos das polinomiais cubicas de hermito na discriminação de formas**. Tese (Doutorado) — [sn], 1996.

ROSENFELD, A. Picture processing by computer. **Proceedings of the IEEE**, v. 54, n. 4, p. 642–653, 1966. Disponível em: <https://ieeexplore.ieee.org/document/1443152>.

RUDIN, W. **Real and Complex Analysis**. 3rd. ed. McGraw-Hill, 1987. Disponível em: <https://www.mheducation.com/highered/product/0070542341.html>.

SHIRLEY, P.; ASHIKHMIN, M.; MARSCHNER, S. **Fundamentals of computer graphics**. [S.l.]: AK Peters/CRC Press, 2009.

TANG, S.; WU, X.; ZHANG, K. On optimal line rasterization. In: KUNII, T. L. (Ed.). **Visual Computing**. Tokyo: Springer Japan, 1992. p. 661–671. ISBN 978-4-431-68204-2.

VINCE, J. Mathematics for computer graphics. In: ____. London: Springer London, 2010. ISBN 978-1-84996-023-6. Disponível em: <https://doi.org/10.1007/978-1-84996-023-6_5>.

YIN, R. K. **Case study research and applications: Design and methods**. 6th. ed. [S.l.]: Sage Publications, 2018.